Even More C++
Outline

- Dynamic Memory
- Data Structures
- Other Data Types
Dynamic Memory

- Fundamental data types take up a fixed size in memory
  - Memory can be allocated when the variables are declared
- There are times when memory size can only be determined at runtime
  - In these cases, programs need to dynamically allocate (and de-allocate) memory
    - This is done using the `new` and `delete` operators
new and new[]

- new is followed by a data type specifier and if there are multiple elements needed, brackets are used, to specify an array

  ```
  pointer = new type
  pointer = new type [number_of_elements]
  ```

- For example:

  ```
  int * foo;
  foo = new int [5];
  ```

- In this example, a pointer to an integer is created, and then a block of memory is allocated to store 5 of them
new and new[]

- So why not just create an array?
- Array size must be declared in one way or another at compile time
- Using dynamic memory assigns memory at runtime so you can use a flexible memory size
- Memory is allocated at runtime from the heap
  - There is no guarantee that there is enough memory to handle a given request
Checking for Allocation Success

- By default, C++ will throw an exception if something went wrong with memory allocation
  - In this case, the program will terminate if the exception is not handled
  - You can tell C++ not to throw an exception and then deal with it in your own code:

```cpp
int * foo;
foo = new (nothrow) int [5];
if (foo == nullptr) {
    // error assigning memory. Take measures.
}
```

- Using exceptions is more efficient – we will talk about those later
delete and delete[]

- **C++ does not handle garbage collection for you**
  - You need to determine when a particular data item is no longer needed and then remove it
  - Use `delete` and `delete[]` to do this
    ```
    delete pointer;
    delete[] pointer;
    ```
  - The “thing” deleted should be either something that was created with `new` or `new[]` before, or it should be a null pointer (in which case nothing happens)
// rememb-o-matic
#include <iostream>
#include <new>
using namespace std;

int main ()
{
    int i,n;
    int * p;
    cout << "How many numbers would you like to type? ";
    cin >> i;
    p= new (nothrow) int[i];
    if (p == nullptr)
        cout << "Error: memory could not be allocated";
    else
    {
        for (n=0; n<i; n++)
        {
            cout << "Enter number: ";
            cin >> p[n];
        }
        cout << "You have entered: ";
        for (n=0; n<i; n++)
            cout << p[n] << ", "
        delete[] p;
    }
    return 0;
}
Dynamic Memory in C

- C++ uses `new` and `delete` to allocate and free memory
- C uses `malloc`, `calloc`, `realloc` and `free`
- Since C++ is built on C, you can still use these functions, but you should not mix them
  - if you use `new` on an item, deallocate it with `delete`
  - if you use `malloc`, `calloc` or `realloc`, deallocate it with `free`
Summary

- Dynamic Memory
- Data Structures
- Other Data Types